2018년 2학기 자료탐색

Tidyverse 패키지란?

Tidyverse는 data science를 위해 개발된 패키지들의 묶음. 여기에 속한 패키지들은 모두 공통된 분석 방식을 공유하고 있으며, 자료 분석에 필요한 모든 과정을 망라하고 있다.

Tidyverse 설치 및 사용하는 방법

```
〉 install.packages("tidyverse")
〉 library(tidyverse)
── Attaching packages ─────────────────────────────────────── tidyverse 1.2.1 ──
√ ggplot2 3.0.0     √ purrr   0.2.5
√ tibble  1.4.2     √ dplyr   0.7.6
√ tidyr   0.8.1     √ stringr 1.3.1
√ readr   1.1.1     √ forcats 0.3.0
── Conflicts ────────────────────────────────────────── tidyverse_conflicts() ──
x dplyr::filter() masks stats::filter()
x dplyr::lag()    masks stats::lag()
```

Core Tidyverse

reader : 자료 불러오기	stringr : 문자열 다루기
tibble : 개선된 형태의 데이터 프레임	forcats : 요인 다루기
tidyr : 분석이 편리한 형태인 tidy 자료 생성	ggplot2 : 자료의 시각화
dplyr : 데이터 프레임 다루기	purr : 함수형 프로그래밍

Tibble의 예

```
> mpg
# A tibble: 234 x 11
   manufacturer model      displ  year   cyl trans      drv     cty   hwy fl    class
   <chr>        <chr>      <dbl> <int> <int> <chr>      <chr> <int> <int> <chr> <chr>
 1 audi         a4           1.8  1999     4 auto(l5)   f        18    29 p     compact
 2 audi         a4           1.8  1999     4 manual(m5) f        21    29 p     compact
 3 audi         a4           2    2008     4 manual(m6) f        20    31 p     compact
 4 audi         a4           2    2008     4 auto(av)   f        21    30 p     compact
 5 audi         a4           2.8  1999     6 auto(l5)   f        16    26 p     compact
 6 audi         a4           2.8  1999     6 manual(m5) f        18    26 p     compact
 7 audi         a4           3.1  2008     6 auto(av)   f        18    27 p     compact
 8 audi         a4 quattro   1.8  1999     4 manual(m5) 4        18    26 p     compact
 9 audi         a4 quattro   1.8  1999     4 auto(l5)   4        16    25 p     compact
10 audi         a4 quattro   2    2008     4 manual(m6) 4        20    28 p     compact
# ... with 224 more rows
```

tibble 형태의 데이터 프레임

```
> MASS::Cars93
   Manufacturer       Model    Type Min.Price Price Max.Price MPG.city MPG.highway             AirBags DriveTrain Cylinders
1         Acura     Integra   Small      12.9  15.9      18.8       25          31                None     Front         4
2         Acura      Legend Midsize      29.2  33.9      38.7       18          25 Driver & Passenger     Front         6
3          Audi          90 Compact      25.9  29.1      32.3       20          26 Driver only        Front         6
4          Audi         100 Midsize      30.8  37.7      44.6       19          26 Driver & Passenger     Front         6
5           BMW        535i Midsize      23.7  30.0      36.2       22          30 Driver only        Rear          4
6         Buick     Century Midsize      14.2  15.7      17.3       22          31 Driver only        Front         4
7         Buick      LeSabre   Large      19.9  20.8      21.7       19          28 Driver only        Front         6
8         Buick   Roadmaster   Large      22.6  23.7      24.9       16          25 Driver only        Rear          6
9         Buick     Riviera Midsize      26.3  26.3      26.3       19          27 Driver only        Front         6
10     Cadillac      DeVille   Large      33.0  34.7      36.3       16          25 Driver only        Front         8
```

전통적인 데이터 프레임

전통적 데이터 프레임을 tibble로 전환

```
〉 as_tibble(cars)
```

개별 벡터를 이요한 tibble의 생성

```
〉 tibble(x=1:3,y=x+1,z=1)
# A tibble: 3 x 3
      x     y     z
  〈int〉〈dbl〉〈dbl〉
1     1     2     1
2     2     3     1
3     3     4     1
```

길이가 1인 스칼라만 순환법칙 적용(만약, z=1:2가 입력되면...?)

```
〉 tibble(x=1:3,y=x+1,z=1:2)
Error: Column `z` must be length 1 or 3, not 2
```

함께 입력되는 변수를 이용한 다른 변수의 생성 가능 (기존 데이터 프레임은 안 됨)

```
〉 data.frame(x=1:3, y=x+1)
Error in data.frame(x = 1:3, y = x + 1) :
    arguments imply differing number of rows: 3, 100
```

문자형 벡터를 요인으로 전환하지 않는다.

```
〉 tibble(x=1:3, y=letters[1:3])
# A tibble: 3 x 2
      x y
  〈int〉〈chr〉
1     1 a
2     2 b
3     3 c
```

행 단위로 입력하여 tibble 생성

```
〉 tribble(
+          ~x,~y,
+           1,"a",
+           2,"b",
+           3,"c"
+ )
```

```
# A tibble: 3 x 2
      x y
  〈dbl〉〈chr〉
1     1 a
2     2 b
3     3 c
```

기존의 데이터 프레임은 요인으로 바꿔주지만,
stringsAsFactors=FALSE 을 넣어주면 요인으로 안 바꿔 준다.

```
〉 str(data.frame(x=1:3, y=letters[1:3]))
'data.frame':       3 obs. of  2 variables:
 $ x: int  1 2 3
 $ y: Factor w/ 3 levels "a","b","c": 1 2 3
〉 str(data.frame(x=1:3, y=letters[1:3], stringsAsFactors=FALSE))
'data.frame':       3 obs. of  2 variables:
 $ x: int  1 2 3
 $ y: chr  "a" "b" "c"
```

Tibble vs 전통적 데이터 프레임

1. 데이터 프레임의 출력 방식
2. 인덱싱 방법
3. row names를 다루는 방식

Tibble vs 전통적 데이터 프레임의 출력방식의 차이

# 전통적 데이터 프레임	# Tibble

가능한 모든 자료를 화면에 출력한다. 대규모 자료의 경우에는 내용 확인에 어렵다.
`〉 MASS::Cars93`
	처음 10개 케이스만 출력 화면의 크기에 따라 출력되는 변수 개수가 조절됨. 한 화면에서 자료의 특성 파악 용이함.
	`〉 as_tibble(MASS::cars93)`
	`# 모든 변수를 보고싶으면 "width=Inf" 를 추가한다.`
	`print(mpg, n=20, width=Inf)`

Tibble vs 전통적 데이터 프레임의 Row names의 처리 방식의 차이

# 전통적 데이터 프레임	# Tibble
자료를 출력할 때 row name 도 함께 출력한다.	자료를 출력할 때 row name 은 숨겨져 있다.

```
> head(mtcars)
                   mpg cyl disp  hp drat    wt  qsec vs am gear carb
Mazda RX4         21.0   6  160 110 3.90 2.620 16.46  0  1    4    4
Mazda RX4 Wag     21.0   6  160 110 3.90 2.875 17.02  0  1    4    4
Datsun 710        22.8   4  108  93 3.85 2.320 18.61  1  1    4    1
Hornet 4 Drive    21.4   6  258 110 3.08 3.215 19.44  1  0    3    1
Hornet Sportabout 18.7   8  360 175 3.15 3.440 17.02  0  0    3    2
Valiant           18.1   6  225 105 2.76 3.460 20.22  1  0    3    1
```

```
> mtcars_t <- as_tibble(mtcars)
> print(mtcars_t, n=6)
# A tibble: 32 x 11
    mpg   cyl  disp    hp  drat    wt  qsec    vs    am  gear  carb
  <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>
1  21       6   160   110  3.9   2.62  16.5     0     1     4     4
2  21       6   160   110  3.9   2.88  17.0     0     1     4     4
3  22.8     4   108    93  3.85  2.32  18.6     1     1     4     1
4  21.4     6   258   110  3.08  3.22  19.4     1     0     3     1
5  18.7     8   360   175  3.15  3.44  17.0     0     0     3     2
6  18.1     6   225   105  2.76  3.46  20.2     1     0     3     1
# ... with 26 more rows
```

Tibble에서 row name 보기

```
# row name을 변수로 생성하여 Tibble에 집어 넣어야 한다.
〉 mtcars_t 〈- rownames_to_column(mtcars_t, var="rowname")
〉 print(mtcars_t, n=5)
```

전통적 데이터 프레임의 인덱싱

```
# 열(변수) 선택
# df[[a]] or df[a] : 벡터 a는 숫자형 또는 문자형
# df[[a]] : 한 변수의 선택, 결과는 벡터
# df[a] : 하나 또는 그 이상의 변수 선택. 결과는 데이터프레임
```

```
〉 df2 〈- data.frame(x=c(2,4,6),y=c("a","b","c"))
〉 df2[1]
  x
1 2
2 4
3 6
〉 df2[[1]]
[1] 2 4 6
```

```
〉 df2["x"]
  x
1 2
2 4
3 6
〉 df2[["x"]]
[1] 2 4 6
〉 df2$x
[1] 2 4 6
```

```
# 행렬의 인덱싱 사용방법
# df[i,j]의 형태
# 선택된 변수가 1개이면 결과는 벡터
# 2개 이상이면 결과는 데이터 프레임
〉 df2[c(1,2),1]
[1] 2 4
〉 df2[c(1,2),]
  x y
1 2 a
2 4 b
```

부분매칭

```
# 기존의 데이터 프레임은 기호 "$" 를 이용하면 부분 매칭이 허용된다.
〉 df1 〈- data.frame(xyz=1:3, abc=letters[1:3])
〉 df1$x
[1] 1 2 3

# Tibble 은 부분 매칭을 허용하지 않는다.
〉 tb1 〈- as_tibble(df1)
〉 tb1$x
NULL
Warning message:
Unknown or uninitialised column: 'x'.
〉 tb1$xyz
[1] 1 2 3
```

Tibble 연습문제

```
〉 df1 〈- tibble(x=1,y=1:9,z=rep(1:3,each=3),w=sample(letters,9))
〉 df1
# A tibble: 9 x 4
      x     y     z w
  〈dbl〉 〈int〉 〈int〉 〈chr〉
1     1     1     1 j
2     1     2     1 b
3     1     3     1 z
4     1     4     2 v
5     1     5     2 u
6     1     6     2 h
7     1     7     3 o
8     1     8     3 m
9     1     9     3 c
```

df1의 두 번째 열 선택하여 벡터로 출력하는 세 가지 인덱싱 방법은?

```
〉 df1$y
[1] 1 2 3 4 5 6 7 8 9
〉 df1[[2]]
[1] 1 2 3 4 5 6 7 8 9
〉 df1[["y"]]
[1] 1 2 3 4 5 6 7 8 9
```

df1의 두 번 째 열의 처음 다섯 개 자료를 행렬의 인덱싱 방법으로 선택

```
〉 df1[c(1:5),2]
```

df1을 전통적인 데이터 프레임으로 전환하고 두 번째 열의 처음 다섯 개 자료 선택. tibble을 대상으로 했을 때와 차이점은?

```
df2 〈- as.data.frame(df1)
df2[1:5,2]
# Tibble은 결과가 Tibble이지만, 전통적은 벡터이다.
```

dplyr 패키지란?

입력된 대부분의 데이터 프레임은 바로 분석할 수 있는 상태가 아니다. 분석에 필요한 적절한 변수가 없을 수도 있으며, 특정 조건을 만족하는 관찰값만을 선택해야 할 수도, 관찰값의 순서를 변경할 수도 있다. 따라서 분석을 할 수 있는 상태로 만들어 줘야 하는데, 이 작업은 매우 필요하나 시간이 많이 소요되는 힘든 작업이다.

dplyr 패키지의 주요 함수

filter() : 특정 조건을 만족하는 관찰값 선택	mutate() : 새로운 변수 생성
arrange() : 특정 변수를 기준으로 관찰값 정렬	group_by() : 그룹 생성
select() : 변수 선택	summarize() : 자료 요약

조건에 따른 관찰값의 선택 : filter()

```
# 기본적인 사용법 : filter(df, condition)
# df : 데이터 프레임
# condition : 관찰값 선택 조건
              비교연산자 : 〉, 〉=, 〈, 〈=, !=, ==
              논리연산자 : &, |, !
              연산자 : %in%
```

변수 mpg가 30 이상인 자동차 선택.

```
〉 mtcars_t 〈- as.tibble(mtcars)
〉 filter(mtcars_t, mpg〉=30)
```

```
# A tibble: 4 x 11
    mpg   cyl  disp    hp  drat    wt  qsec    vs    am  gear  carb
  <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>
1  32.4     4  78.7    66  4.08  2.2   19.5     1     1     4     1
2  30.4     4  75.7    52  4.93  1.62  18.5     1     1     4     2
3  33.9     4  71.1    65  4.22  1.84  19.9     1     1     4     1
4  30.4     4  95.1   113  3.77  1.51  16.9     1     1     5     2
```

변수 mpg가 30 이상, 변수 wt가 1.8 미만인 자동차 선택

```
〉 filter(mtcars_t, mpg〉=30 & wt〈1.8 )
```

```
# A tibble: 2 x 11
    mpg   cyl  disp    hp  drat    wt  qsec    vs    am  gear  carb
  <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>
1  30.4     4  75.7    52  4.93  1.62  18.5     1     1     4     2
2  30.4     4  95.1   113  3.77  1.51  16.9     1     1     5     2
```

변수 mpg가 30 이하이고, 변수 cyl은 6 또는 8, 변수 am은 1 인 자동차 선택

```
〉 filter(mtcars_t, mpg 〈= 30 & cyl %in% c(6,8) & am == 1)
```

```
# A tibble: 5 x 11
    mpg   cyl  disp    hp  drat    wt  qsec    vs    am  gear  carb
  <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>
1  21        6   160   110  3.9   2.62  16.5     0     1     4     4
2  21        6   160   110  3.9   2.88  17.0     0     1     4     4
3  15.8      8   351   264  4.22  3.17  14.5     0     1     5     4
4  19.7      6   145   175  3.62  2.77  15.5     0     1     5     6
5  15        8   301   335  3.54  3.57  14.6     0     1     5     8
```

변수 mpg의 값이 mpg의 중앙값과 Q3(제3사분위수) 사이에 있는 자동차를 선택.

```
〉 filter(mtcars_t, mpg 〉= median(mpg) & mpg 〈= quantile(mpg,probs=0.75))
```

```
# A tibble: 10 x 11
    mpg   cyl  disp    hp  drat    wt  qsec    vs    am  gear  carb
  <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>
 1  21        6   160   110  3.9   2.62  16.5     0     1     4     4
 2  21        6   160   110  3.9   2.88  17.0     0     1     4     4
 3  22.8      4   108    93  3.85  2.32  18.6     1     1     4     1
 4  21.4      6   258   110  3.08  3.22  19.4     1     0     3     1
 5  22.8      4   141.   95  3.92  3.15  22.9     1     0     4     2
 6  19.2      6   168.  123  3.92  3.44  18.3     1     0     4     4
 7  21.5      4   120.   97  3.7   2.46  20.0     1     0     3     1
 8  19.2      8   400   175  3.08  3.84  17.0     0     0     3     2
 9  19.7      6   145   175  3.62  2.77  15.5     0     1     5     6
10  21.4      4   121   109  4.11  2.78  18.6     1     1     4     2
```

벡터가 특정 두 숫자 사이에 있는지를 확인하는 방법 : x>=left & x<=right or between(x,left,right)

```
> filter(mtcars_t, between(mpg, median(mpg), quantile(mpg,probs=0.75)))
# A tibble: 10 x 11
     mpg   cyl  disp    hp  drat    wt  qsec    vs    am  gear  carb
   <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>
 1  21.0     6  160.   110  3.90  2.62  16.5     0     1     4     4
 2  21.0     6  160.   110  3.90  2.88  17.0     0     1     4     4
 3  22.8     4  108.    93  3.85  2.32  18.6     1     1     4     1
 4  21.4     6  258.   110  3.08  3.22  19.4     1     0     3     1
 5  22.8     4  141.    95  3.92  3.15  22.9     1     0     4     2
 6  19.2     6  168.   123  3.92  3.44  18.3     1     0     4     4
 7  21.5     4  120.    97  3.70  2.46  20.0     1     0     3     1
 8  19.2     8  400.   175  3.08  3.84  17.0     0     0     3     2
 9  19.7     6  145.   175  3.62  2.77  15.5     0     1     5     6
10  21.4     4  121.   109  4.11  2.78  18.6     1     1     4     2
```

airquality 데이터 프레임의 변수 Ozone 또는 solar.R이 결측값인 관찰값 선택

```
> airs <- as.tibble(airquality)
> filter(airs, is.na(Ozone) | is.na(Solar.R))
# A tibble: 42 x 6
   Ozone Solar.R  Wind  Temp Month   Day
   <int>   <int> <dbl> <int> <int> <int>
 1    NA      NA  14.3    56     5     5
 2    28      NA  14.9    66     5     6
 3    NA     194   8.6    69     5    10
 4     7      NA   6.9    74     5    11
 5    NA      66  16.6    57     5    25
 6    NA     266  14.9    58     5    26
 7    NA      NA   8.0    57     5    27
 8    NA     286   8.6    78     6     1
 9    NA     287   9.7    74     6     2
10    NA     242  16.1    67     6     3
# ... with 32 more rows
```

관찰값의 정렬 : filter()

```
# 기본적인 사용법 : arrange(df, var_1, var_2, ... )
# df : 데이터 프레임
# var_1 : 제1 정렬 기준 변수
# var_2 : var_1 의 값이 같은 관찰값의 정렬 기준
# 오름차순이 디폴트값, desc()를 사용하면 내림차순 정렬
```

연비(mpg)가 제일 안좋은 차부터 재배열

```
> mtcars_t <- as_tibble(mtcars)
> arrange(mtcars_t, mpg)
# A tibble: 32 x 11
     mpg   cyl  disp    hp  drat    wt  qsec    vs    am  gear  carb
   <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>
 1  10.4     8  472.   205  2.93  5.25  18.0     0     0     3     4
 2  10.4     8  460.   215  3.00  5.42  17.8     0     0     3     4
 3  13.3     8  350.   245  3.73  3.84  15.4     0     0     3     4
 4  14.3     8  360.   245  3.21  3.57  15.8     0     0     3     4
 5  14.7     8  440.   230  3.23  5.34  17.4     0     0     3     4
 6  15.0     8  301.   335  3.54  3.57  14.6     0     1     5     8
 7  15.2     8  276.   180  3.07  3.78  18.0     0     0     3     3
 8  15.2     8  304.   150  3.15  3.44  17.3     0     0     3     2
 9  15.5     8  318.   150  2.76  3.52  16.9     0     0     3     2
10  15.8     8  351.   264  4.22  3.17  14.5     0     1     5     4
# ... with 22 more rows
```

mpg의 값이 가장 좋지 않은 자동차부터 배열하되, mpg의 값이 같은 경우에는 wt 값이 높은 자동차부터 배열.

```
> arrange(mtcars_t, mpg, desc(wt))
# A tibble: 32 x 11
     mpg   cyl  disp    hp  drat    wt  qsec    vs    am  gear  carb
   <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>
 1  10.4     8  460.   215  3.00  5.42  17.8     0     0     3     4
 2  10.4     8  472.   205  2.93  5.25  18.0     0     0     3     4
 3  13.3     8  350.   245  3.73  3.84  15.4     0     0     3     4
 4  14.3     8  360.   245  3.21  3.57  15.8     0     0     3     4
 5  14.7     8  440.   230  3.23  5.34  17.4     0     0     3     4
 6  15.0     8  301.   335  3.54  3.57  14.6     0     1     5     8
 7  15.2     8  276.   180  3.07  3.78  18.0     0     0     3     3
 8  15.2     8  304.   150  3.15  3.44  17.3     0     0     3     2
 9  15.5     8  318.   150  2.76  3.52  16.9     0     0     3     2
10  15.8     8  351.   264  4.22  3.17  14.5     0     1     5     4
# ... with 22 more rows
```

5월 1일 ~ 5월 10일 사이의 자료만을 대상으로 Ozone 값이 가장 낮았던 날부터 다시 재배열

```
airs_1 <- arrange(filter(airs, Month==5, Day<=10), Ozone)
```

```
# A tibble: 10 x 6
   Ozone Solar.R  Wind  Temp Month   Day
   <int>   <int> <dbl> <int> <int> <int>
 1     8      19  20.1    61     5     9
 2    12     149  12.6    74     5     3
 3    18     313  11.5    62     5     4
 4    19      99  13.8    59     5     8
 5    23     299   8.6    65     5     7
 6    28      NA  14.9    66     5     6
 7    36     118   8      72     5     2
 8    41     190   7.4    67     5     1
 9    NA      NA  14.3    56     5     5
10    NA     194   8.6    69     5    10
```

iars_1에서 Ozone이 결측값인 케이스를 가장 앞으로 배열

```
> arrange(airs_1, !is.na(Ozone))
```

```
# A tibble: 10 x 6
   Ozone Solar.R  Wind  Temp Month   Day
   <int>   <int> <dbl> <int> <int> <int>
 1    NA      NA  14.3    56     5     5
 2    NA     194   8.6    69     5    10
 3     8      19  20.1    61     5     9
 4    12     149  12.6    74     5     3
 5    18     313  11.5    62     5     4
 6    19      99  13.8    59     5     8
 7    23     299   8.6    65     5     7
 8    28      NA  14.9    66     5     6
 9    36     118   8      72     5     2
10    41     190   7.4    67     5     1
```

airs_1을 Ozone의 값이 가장 높은 날부터 다시 배열 하되, 결측값을 가장 앞으로 배열

```
> arrange(airs_1, !is.na(Ozone), desc(Ozone))
```

```
# A tibble: 10 x 6
   Ozone Solar.R  Wind  Temp Month   Day
   <int>   <int> <dbl> <int> <int> <int>
 1    NA      NA  14.3    56     5     5
 2    NA     194   8.6    69     5    10
 3    41     190   7.4    67     5     1
 4    36     118   8      72     5     2
 5    28      NA  14.9    66     5     6
 6    23     299   8.6    65     5     7
 7    19      99  13.8    59     5     8
 8    18     313  11.5    62     5     4
 9    12     149  12.6    74     5     3
10     8      19  20.1    61     5     9
```

변수의 선택 : select()

데이터 세트의 크기가 증가하여 변수의 개수가 수백 또는 수천이 되는 경우가 발생한다. 이럴 경우 분석에 필요한 변수 선택으로 데이터 세트의 크기를 감소시킬 필요가 있다.
기본적인 사용법 : select(df, 변수이름 or 문자열 매칭 함수)

mtcars에서 row names를 변수 rowname으로 전환하고, 변수 rowname과 mpg 선택

```
> mtcars_t <- as_tibble(mtcars)
> mtcars_t <-  rownames_to_column(mtcars_t, var="rowname")
> select(mtcars_t, rowname, mpg)
```

```
# A tibble: 32 x 2
   rowname             mpg
   <chr>             <dbl>
 1 Mazda RX4          21
 2 Mazda RX4 Wag      21
 3 Datsun 710         22.8
 4 Hornet 4 Drive     21.4
 5 Hornet Sportabout  18.7
 6 Valiant            18.1
 7 Duster 360         14.3
 8 Merc 240D          24.4
 9 Merc 230           22.8
10 Merc 280           19.2
# ... with 22 more rows
```

mtcars_t 의 첫 번째 변수 mpg 부터 여섯 번째 변수 wt 까지 모두 선택

```
〉 select(mtcars_t,mpg:wt)
〉 select(mtcars_t,c(2:7))
```

```
# A tibble: 32 x 6
     mpg   cyl  disp    hp  drat    wt
   <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>
 1  21       6  160    110  3.9   2.62
 2  21       6  160    110  3.9   2.88
 3  22.8     4  108     93  3.85  2.32
 4  21.4     6  258    110  3.08  3.22
 5  18.7     8  360    175  3.15  3.44
 6  18.1     6  225    105  2.76  3.46
 7  14.3     8  360    245  3.21  3.57
 8  24.4     4  147.    62  3.69  3.19
 9  22.8     4  141.    95  3.92  3.15
10  19.2     6  168.   123  3.92  3.44
# ... with 22 more rows
```

mtcars_t 에서 변수 rowname 과 qsec 에서 carb 까지 제거

```
〉 select(mtcars_t, -(rowname),-(qsec:carb))
```

```
# A tibble: 32 x 6
     mpg   cyl  disp    hp  drat    wt
   <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>
 1  21       6  160    110  3.9   2.62
 2  21       6  160    110  3.9   2.88
 3  22.8     4  108     93  3.85  2.32
 4  21.4     6  258    110  3.08  3.22
 5  18.7     8  360    175  3.15  3.44
 6  18.1     6  225    105  2.76  3.46
 7  14.3     8  360    245  3.21  3.57
 8  24.4     4  147.    62  3.69  3.19
 9  22.8     4  141.    95  3.92  3.15
10  19.2     6  168.   123  3.92  3.44
# ... with 22 more rows
```

문자열 매칭 함수

```
# 변수가 많은 경우 선택하고자 하는 변수의 이름을 일일이 나열하는 것은 비효율적이다. 따라서 문자열 매칭 함수를 사용한다.
# start_with( “x” ) : 이름이 “x” 로 시작하는 변수 선택
# ends_with( “x” ) : 이름이 “x” 로 끝나는 변수 선택
# contians_with( “x” ) : 이름에 “x” 가 포함되는 변수 선택
# num_range( “x” , 1:10) : x1, x2, x3, ... , x10 을 선택
```

데이터 프레임 mtcars_t 에서

“m” 으로 시작되는 변수 선택

```
〉 select(mtcars_t, starts_with("m"))
```

```
# A tibble: 32 x 1
     mpg
   <dbl>
 1  21
 2  21
 3  22.8
 4  21.4
 5  18.7
 6  18.1
 7  14.3
 8  24.4
 9  22.8
10  19.2
# ... with 22 more rows
```

“p” 로 끝나는 변수 선택

```
〉 select(mtcars_t, ends_with("p"))
```

```
# A tibble: 32 x 2
    disp    hp
   <dbl> <dbl>
 1  160    110
 2  160    110
 3  108     93
 4  258    110
 5  360    175
 6  225    105
 7  360    245
 8  147.    62
 9  141.    95
10  168.   123
# ... with 22 more rows
```

“a” 가 이름에 있는 변수 선택

```
〉 select(mtcars_t, contains("a"))
```

```
# A tibble: 32 x 5
   rowname           drat    am  gear  carb
   <chr>            <dbl> <dbl> <dbl> <dbl>
 1 Mazda RX4         3.9      1     4     4
 2 Mazda RX4 Wag     3.9      1     4     4
 3 Datsun 710        3.85     1     4     1
 4 Hornet 4 Drive    3.08     0     3     1
 5 Hornet Sportabout 3.15     0     3     2
 6 Valiant           2.76     0     3     1
 7 Duster 360        3.21     0     3     4
 8 Merc 240D         3.69     0     4     2
 9 Merc 230          3.92     0     4     2
10 Merc 280          3.92     0     4     4
# ... with 22 more rows
```

예제 데이터 Iris

```
# 3종류의 붓꽃 각 50송이 씩 총 150송이의 붓꽃의 꽃받침 길이와 폭, 꽃잎의 길이와 폭을 측정한 데이터다.
# 각 변수 명 : Sepal.Length, Sepal.Width, Petal.Length, Petal.Width, Species
```

```
# A tibble: 5 x 5
  Sepal.Length Sepal.Width Petal.Length Petal.Width Species
         <dbl>       <dbl>        <dbl>       <dbl> <fct>
1          5.1         3.5          1.4         0.2 setosa
2          4.9         3            1.4         0.2 setosa
```

데이터 프레임 Iris에서 (단, ignore.case=FALSE를 사용하면 대소문자가 구분된다.)

“pe” 가 이름에 있는 변수 선택

```
〉 select(iris,contains("pe", ignore.case=FALSE))
```

```
# A tibble: 150 x 1
   Species
   <fct>
 1 setosa
 2 setosa
 3 setosa
 4 setosa
 5 setosa
 6 setosa
 7 setosa
 8 setosa
 9 setosa
10 setosa
# ... with 140 more rows
```

“pe” 가 이름에 있는 변수 제거

```
〉 select(iris,-(contains("pe", ignore.case=FALSE)))
```

```
# A tibble: 150 x 4
   Sepal.Length Sepal.Width Petal.Length Petal.Width
          <dbl>       <dbl>        <dbl>       <dbl>
 1          5.1         3.5          1.4         0.2
 2          4.9         3            1.4         0.2
 3          4.7         3.2          1.3         0.2
 4          4.6         3.1          1.5         0.2
 5          5           3.6          1.4         0.2
 6          5.4         3.9          1.7         0.4
 7          4.6         3.4          1.4         0.3
 8          5           3.4          1.5         0.2
 9          4.4         2.9          1.4         0.2
10          4.9         3.1          1.5         0.1
# ... with 140 more rows
```

함수 everything()

몇몇 변수를 제일 앞으로 옮겨서 다시 배치할 때 사용되는 함수이다.

```
〉 select(iris, Species, everything())
```

```
# A tibble: 150 x 5
   Species Sepal.Length Sepal.Width Petal.Length Petal.Width
   <fct>          <dbl>       <dbl>        <dbl>       <dbl>
 1 setosa           5.1         3.5          1.4         0.2
 2 setosa           4.9         3            1.4         0.2
 3 setosa           4.7         3.2          1.3         0.2
 4 setosa           4.6         3.1          1.5         0.2
 5 setosa           5           3.6          1.4         0.2
 6 setosa           5.4         3.9          1.7         0.4
 7 setosa           4.6         3.4          1.4         0.3
 8 setosa           5           3.4          1.5         0.2
 9 setosa           4.4         2.9          1.4         0.2
10 setosa           4.9         3.1          1.5         0.1
# ... with 140 more rows
```

변수 이름 변경

select() 로 변수 이름을 변경 –〉 선언하지 않은 변수들이 자동으로 제거(비효율적).

함수 rename() 이 변수 이름 변경에는 더 효과적이다.

```
> select(mtcars_t, Model=rowname)
# A tibble: 32 x 1
   Model
   <chr>
 1 Mazda RX4
 2 Mazda RX4 Wag
 3 Datsun 710
 4 Hornet 4 Drive
 5 Hornet Sportabout
 6 Valiant
 7 Duster 360
 8 Merc 240D
 9 Merc 230
10 Merc 280
# ... with 22 more rows
```

```
> rename(mtcars_t, Model=rowname)
# A tibble: 32 x 12
   Model              mpg   cyl  disp    hp  drat    wt  qsec    vs    am  gear  carb
   <chr>            <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>
 1 Mazda RX4         21       6  160    110  3.9   2.62  16.5     0     1     4     4
 2 Mazda RX4 Wag     21       6  160    110  3.9   2.88  17.0     0     1     4     4
 3 Datsun 710        22.8     4  108     93  3.85  2.32  18.6     1     1     4     1
 4 Hornet 4 Drive    21.4     6  258    110  3.08  3.22  19.4     1     0     3     1
 5 Hornet Sportabout 18.7     8  360    175  3.15  3.44  17.0     0     0     3     2
 6 Valiant           18.1     6  225    105  2.76  3.46  20.2     1     0     3     1
 7 Duster 360        14.3     8  360    245  3.21  3.57  15.8     0     0     3     4
 8 Merc 240D         24.4     4  147.    62  3.69  3.19  20       1     0     4     2
 9 Merc 230          22.8     4  141.    95  3.92  3.15  22.9     1     0     4     2
10 Merc 280          19.2     6  168.   123  3.92  3.44  18.3     1     0     4     4
# ... with 22 more rows
```

새로운 변수의 추가 : mutate()

데이터 프레임을 구성하고 있는 변수를 이용하여 새로운 변수를 생성

생성된 변수는 데이터 프레임의 마지막 변수로 추가된다.

기본적인 사용법 : mutate(df, 새로운 변수 생성 표현식)

데이터 프레임 mtcars_t 에서 kml 과 gp_kml을 생성하고 데이터 프레임의 처음 두 변수로 추가

kml : 1 mpg는 0.43 kml

gp_kml : kml이 10 이상이면 “good”, 10 미만이면 “bad”

```
〉 mtcars_t 〈– mutate(mtcars_t, kml=0.43*mpg, gp_kml=if_else(kml〉=10,"good","bad"))
〉 select(mtcars_t, kml, gp_kml, everything())
```

```
# A tibble: 32 x 14
     kml gp_kml rowname            mpg   cyl  disp    hp  drat    wt  qsec    vs    am  gear  carb
   <dbl> <chr>  <chr>            <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>
 1  9.03 bad    Mazda RX4         21       6  160    110  3.9   2.62  16.5     0     1     4     4
 2  9.03 bad    Mazda RX4 Wag     21       6  160    110  3.9   2.88  17.0     0     1     4     4
 3  9.80 bad    Datsun 710        22.8     4  108     93  3.85  2.32  18.6     1     1     4     1
 4  9.20 bad    Hornet 4 Drive    21.4     6  258    110  3.08  3.22  19.4     1     0     3     1
 5  8.04 bad    Hornet Sportabout 18.7     8  360    175  3.15  3.44  17.0     0     0     3     2
 6  7.78 bad    Valiant           18.1     6  225    105  2.76  3.46  20.2     1     0     3     1
 7  6.15 bad    Duster 360        14.3     8  360    245  3.21  3.57  15.8     0     0     3     4
 8 10.5  good   Merc 240D         24.4     4  147.    62  3.69  3.19  20       1     0     4     2
 9  9.80 bad    Merc 230          22.8     4  141.    95  3.92  3.15  22.9     1     0     4     2
10  8.26 bad    Merc 280          19.2     6  168.   123  3.92  3.44  18.3     1     0     4     4
# ... with 22 more rows
```

만약 새로운 변수만 남기고 나머지는 제거하고 싶다면 함수 transmute() 를 사용한다.

```
〉 transmute(mtcars_t, kml=mpg*0.43, gp_kml=if_else(kml〉=10,"good","bad"))
# A tibble: 32 x 2
     kml gp_kml
   <dbl> <chr>
 1  9.03 bad
 2  9.03 bad
 3  9.80 bad
 4  9.20 bad
 5  8.04 bad
 6  7.78 bad
 7  6.15 bad
 8 10.5  good
 9  9.80 bad
10  8.26 bad
# ... with 22 more rows
```

그룹 생성 및 그룹별 자료 요약 : group_by, summarize()

```
# 함수 summarzie() 는 변수의 요약 통계량을 계산해준다.
# 기본적인 사용법 : summarize(df, name=fun)
# 결과는 tibble로 저장되며, name 은 계산된 통계량 값의 이름이다.
# 반드시 결과가 숫자 하나로 출력되는 함수만 사용 가능하다. ex : mean(), sd(), min(), max()
# 결과가 벡터인 함수는 사용 불가능하다. ex : range()
# n() : 케이스의 개수 / n_distinct() : 서로 다른 숫자의 개수
〉 summarize(mpg, n=n(), n_hwy=n_distinct(hwy), avg_hwy=mean(hwy), sd_hwy=sd(hwy))
# A tibble: 1 x 4
      n     n_hwy   avg_hwy   sd_hwy
  〈int〉   〈int〉     〈dbl〉    〈dbl〉
1    234    27       23.4      5.95
```

함수 group_by()

```
# 한 개 이상의 변수로 데이터 프레임을 그룹으로 구분
# 기본적인 사용법 : group_by(df, var)
# 실행결과 : group_df 라는 class 속성이 추가된 tibble. 출력된 상태로는 실행 전과 차이가 없다.
〉 by_cyl 〈- group_by(mpg,cyl)
〉 summarize(by_cyl, n=n(), avg_mpg=mean(hwy))
# A tibble: 4 x 3
    cyl     n   avg_mpg
  〈int〉 〈int〉    〈dbl〉
1     4    81     28.8
2     5     4     28.8
3     6    79     22.8
4     8    70     17.6
```

Pipe 기능

```
# 한 명령문의 결과물을 바로 다음 명령문의 입력요소로 직접 사용할 수 있도록 명령문들을 서로 연결하는 기능을 의미한다.
# 이것의 장점은 분석 중간 단계에 생성되는 많은 객체들을 따로 저장할 필요가 없기 때문에 프로그래밍이 매우 깔끔해지며,
# 분석 흐름을 쉽게 이해할 수 있다.
# 기본적인 형태 : lhs %〉% rhs
# lhs : 객체
# rhs : lhs를 입력요소로 하는 함수
# 예를들어 x %〉% f 는 객체 x를 함수 f()의 입력요소로 하는 f(x)를 의미한다.
# 만일 rhs에 다른 요소가 있다면 lhs는 rhs의 첫 번째 입력요소가 된다.
# 따라서 x %〉% f(y) 는 f(x,y,)를 의미한다.
# lhs가 rhs의 첫 번째 요소가 아닌 경우에는 원하는 위치에 마침표(.)를 찍어야한다
# x %〉% f(y,.) : f(y,x)
```

데이터 프레임 mpg를 변수 cyl 에 따라 그룹으로 구분하고, 각 그룹에 속한 케이스의 개수 및 각 그룹별 변수 hwy 의 평균값 계산

```
〉 mpg %〉% group_by(cyl) %〉% summarize(n=n(), hwy_mean=mean(hwy))
# A tibble: 4 x 3
    cyl     n    hwy_mean
  〈int〉 〈int〉     〈dbl〉
1     4    81     28.8
2     5     4     28.8
3     6    79     22.8
4     8    70     17.6
```

데이터 프레임 airquality 에서

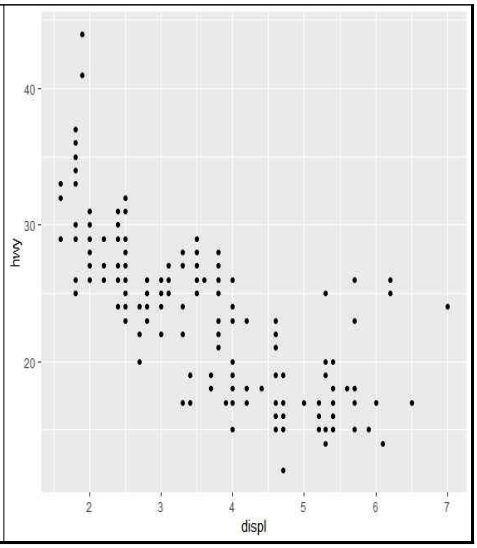
```
# 1. 월별 변수 Ozone의 평균값
〉 air_Month 〈- airquality %〉% as_tibble() %〉% group_by(Month)

# 2. 월별 날수, 변수 Ozone에 결측값이 있는 날수 및 실제 측정이 된 날수
〉 air_Month %〉% summarise(Oz_mean=mean(Ozone, na.rm=TRUE))

# 3. 월별 첫날과 마지막 날 변수 Ozone의 값
〉 air_Month %〉% summarise(n_total=n(), n_miss=sum(is.na(Ozone)), n_obs=sum(!is.na(Ozone)))
# 4. 월별 변수 Ozone의 가장 작은 값과 가장 큰 값

〉 air_Month %〉% summarise(n_total=n(), Max_Oz=max(Ozone,na.rm=TRUE), Min_Oz=min(Ozone,na.rm=TRUE))
# 5. 월별로 변수 Ozone의 개별 값이 전체 기간 동안의 평균값보다 작은 날수

〉 m_Oz = mean(with(airquality,Ozone), na.rm=TRUE)
〉 air_Month %〉% summarise(low_Oz=sum(Ozone〈m_Oz, na.rm=TRUE))
```

n 번째 값 추출하기

```
〉 x 〈- c(2,4,6,8,10)
〉 x
[1]  2  4  6  8 10

〉 first(x)
[1] 2
〉 last(x)
[1] 10
〉 first(x, order_by=order(-x))
[1] 10
〉 nth(x,2)
[1] 4
〉 nth(x,-2)
[1] 8
〉 nth(x,6)
[1] NA
〉 nth(x,6,default=99)
[1] 99
```

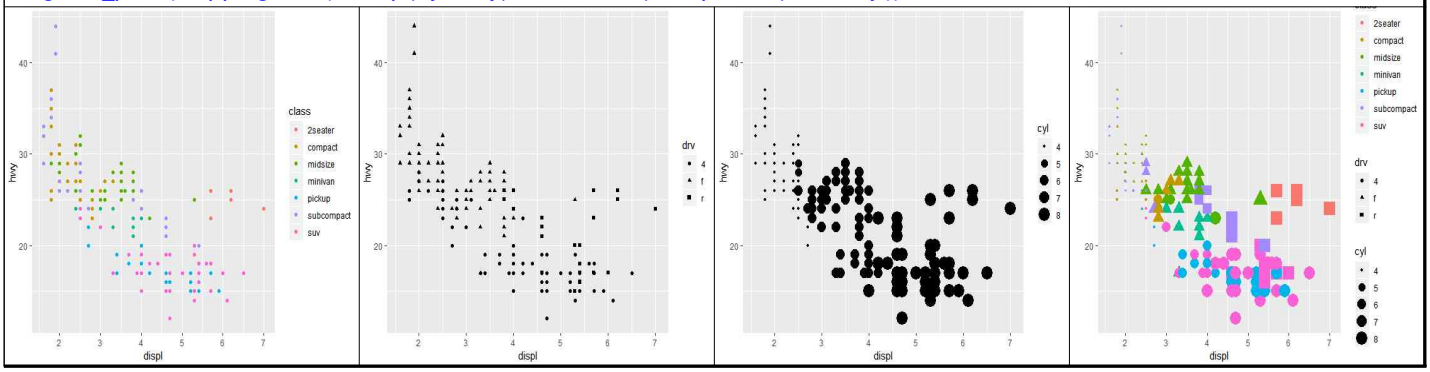
패키지 ggplot2에 있는 데이터 프레임 mpg의 변수 displ과 hwy의 산점도 작성

```
# 함수 ggplot() : 데이터 프레임 data 지정. 그래프가 작성될 비어있는 좌표계 작성
# geom_point() : 실질적인 그래프, 레이어를 작성하는 geom 함수 중 하나
# mapping : geom 함수 내에서 함수 aes()와 함께 데이터와 시각적 요소를 서로 연결

# 반드시 필요한 3가지
# ggplot(data = 〈Data〉) +
#    〈Geom_Function〉(mapping = aes(〈Mappings〉))

# mapping과 setting
# mapping : 데이터 값과 연결되어 결정. 함수 aes() 안에서 연
# setting : 사용자가 일정한 값을 지정

> ggplot(data=mpg) +
+    geom_point(mapping=aes(x=displ, y=hwy))
```



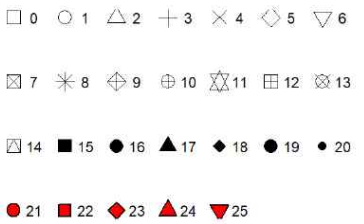
패키지 ggplot2에 있는 데이터 프레임 mpg의 변수 displ과 hwy의 산점도 작성

```
# 산점도에 색 변경
ggplot(data=mpg) +
  geom_point(mapping=aes(x=displ, y=hwy, color=class))

# 산점도의 점의 모양 변경
ggplot(data=mpg) +
  geom_point(mapping=aes(x=displ, y=hwy, shape=drv))

# 산점도의 점의 크기를 변경
ggplot(data=mpg) +
  geom_point(mapping=aes(x=displ, y=hwy, size=cyl))

# 산점도의 점의 모양과 색과 크기를 동시에 변경
ggplot(data=mpg) +
  geom_point(mapping=aes(x=displ, y=hwy, color=class, shape=drv, size=cyl))
```



시각적 요소의 setting

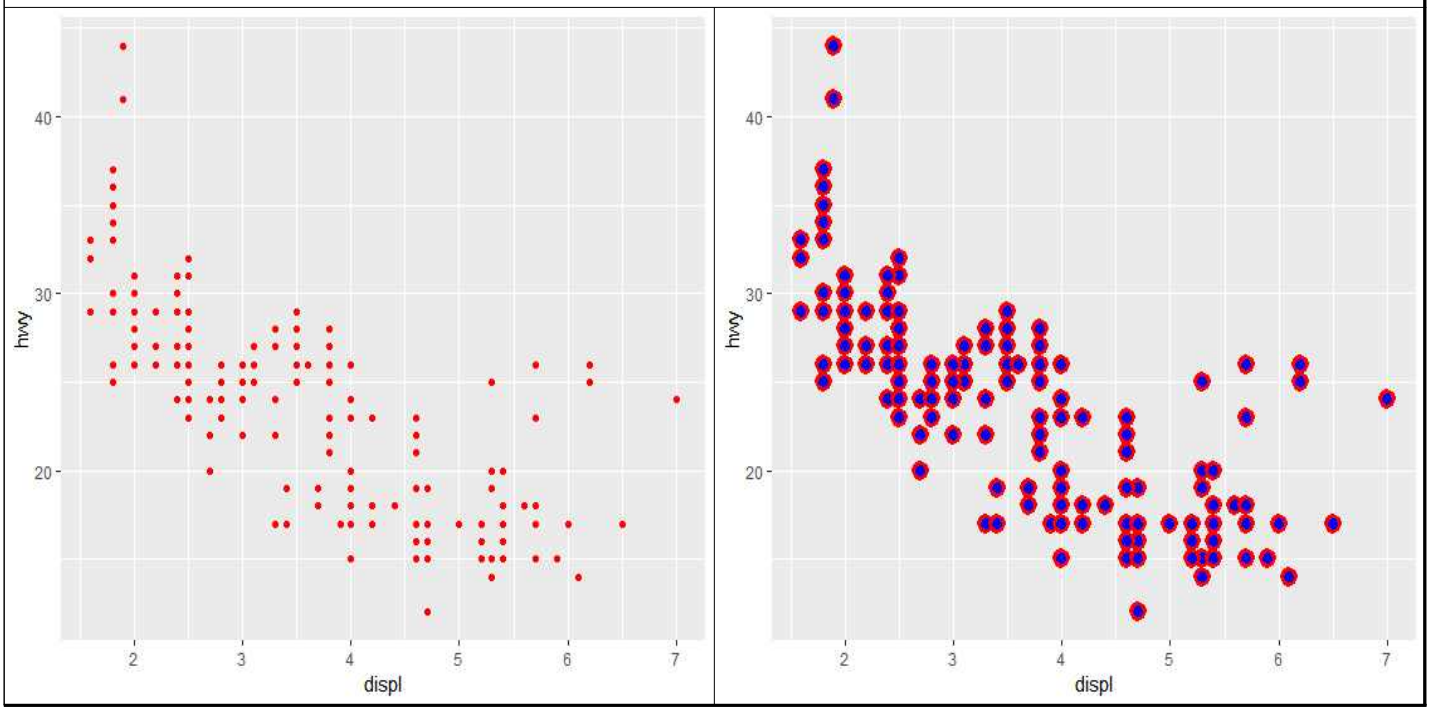
```
# 함수 aes() 밖에서 사용자가 원하는 값으로 지정
# geom 함수의 입력 요소가 된다.

# 시각적 요소 color, size, shape에 값 지정 법칙
(1) color : 색깔을 나타내는 문자열 지정
(2) size : 점 크기를 mm 단위로 지정
(3) shape : 점의 형태를 나타내는 0~26 사이의 숫자
```



도형에 색깔 지정 방법
(1) 0~14의 외각선 및 15~20의 도형 색 : color 사용
(2) 21~25의 외각선 : color 사용
(3) 21~25의 내부 색 : fill 사용

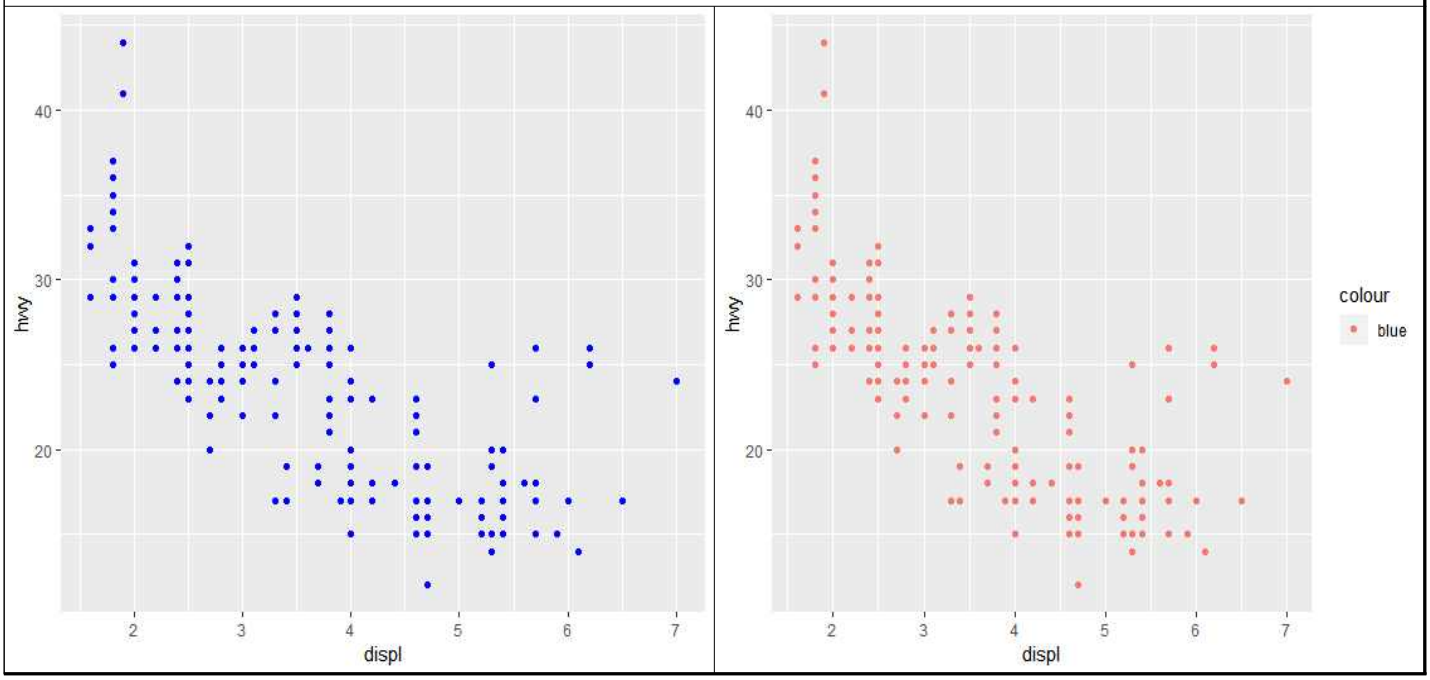
시각적 요소의 setting

```
# 사용자가 임의로 지정 -> 모든 점을 빨간색으로...(사용자 지정)
> ggplot(data=mpg) +
+    geom_point(mapping=aes(x=displ, y=hwy), color="red")

# 여러 요소를 동시에 setting
> ggplot(data=mpg) +
+    geom_point(mapping=aes(x=displ, y=hwy), color="red", shape=21, size=3, fill="blue", stroke=2)
```



함수 aes() 안에 시각적 요소에 특정값을 setting한 경우의 결과

```
# setting
> ggplot(data=mpg) +
+    geom_point(mapping=aes(x=displ, y=hwy),color="blue")
# mapping
> ggplot(data=mpg) +
+    geom_point(mapping=aes(x=displ, y=hwy, color="blue"))
```

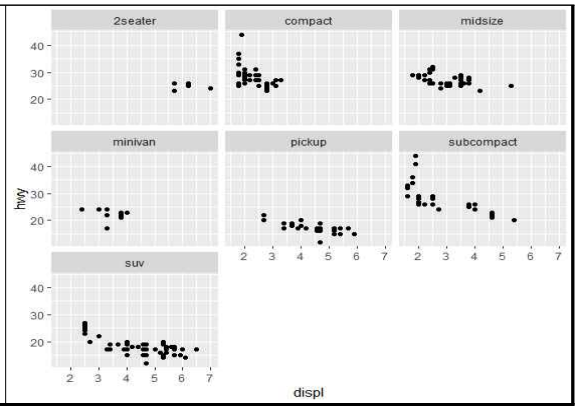


그룹별 그래프 작성 : Facet

```
# 범주형 변수가 다른 변수에 미치는 영향력을 그래프로 확인하는 방법
1) 시각적 요소에 범주형 변수를 mapping
2) 범주형 변수로 그룹 구성하고, 각 그룹별 그래프 작성 : faceting

# Facet을 적용하기 위한 함수
1) facet_wrap() : 한 변수에 의한 facet
2) facet_grid() : 한 변수 또는 두 변수에 의한 facet
```

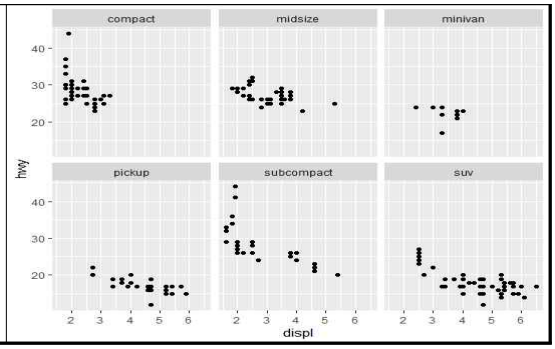
함수 facet_wrap()에 의한 faceting

```
# 데이터를 구분하는 변수가 하나인 경우 : facet_wrap(~x)

# 데이터 프레임 mpg의 변수 displ과 hwy의 산점도를 class의 범주별로 작성
> ggplot(data=mpg) +
+    geom_point(mapping=aes(x=displ,y=hwy))+
+    facet_wrap(~class)

# 패널 "2seater"에는 적은 수의 데이터 존재
# class가 "2seater"인 케이스 제거 후 다시 작성
```



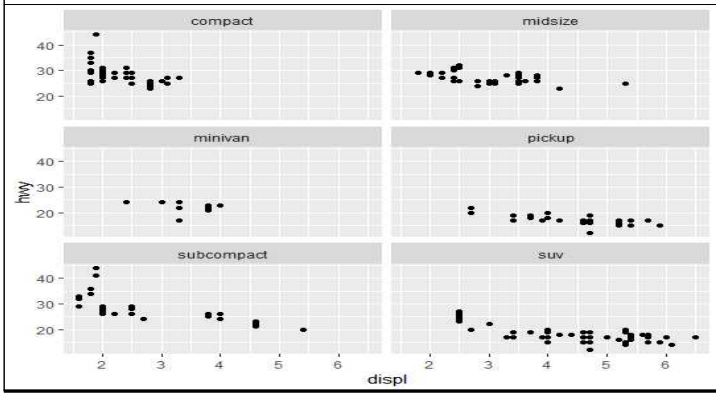
데이터 프레임 mpg의 변수 displ과 hwy의 산점도를 class의 범주별로 작성 (2seater 케이스 제외)

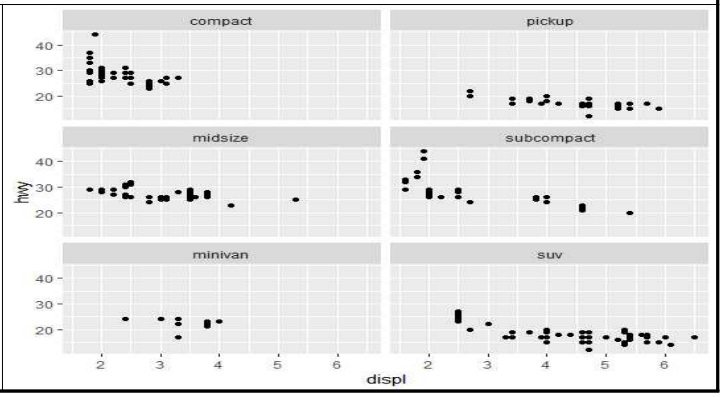
```
# 패널 배치 조절
 - 열의 수 지정 : ncol=
 - 행의 수 지정 : nrow=
 - 배치 순서 지정 : 행단위(디폴트) / 열단위(dir= "v" )

> mpg %>% filter(class!="2seater") %>% ggplot(mapping=aes(x=displ,y=hwy))
+
+    geom_point() +
+    facet_wrap(~class)
```



2x3 패널 배치를 3x2 배치로 수정 : ncol=2

```
# 객체 생성
> pp <- ggplot(data=filter(mpg, class!="2seater")) +
+    geom_point(mapping=aes(x=displ,y=hwy))
# 행으로 배치
> pp + facet_wrap(~class, ncol=2)
# 열로 배치
> pp + facet_wrap(~class, ncol=2, dir="v")
```



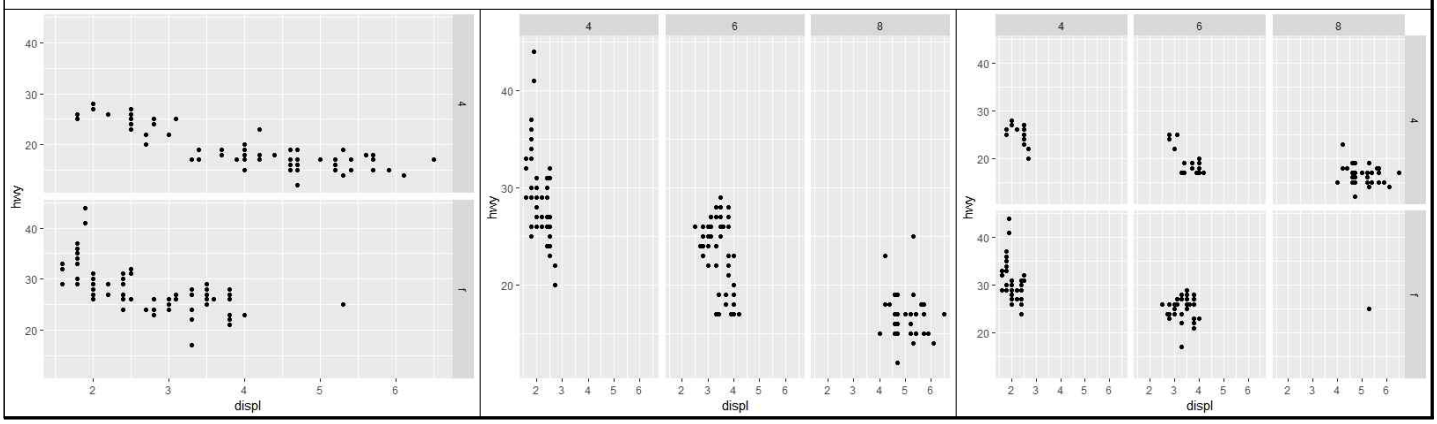


함수 facet_grid()에 의한 faceting

```
# 한 변수에 의한 faceting
 – 하나의 행으로 패널 배치 : facet_grid(.~x)
 – 하나의 열로 패널 배치 : facet_gird(x~.)

# 두 변수에 의한 faceting : facet_grid(y~x)
 – 행범주 : 변수 y의 범주
 – 열범주 : 변수 x의 범주
```

mpg에서 데이터를 변수 drv와 cyl로 구분하여 displ과 hwy의 산점도를 작성. 단, drv가 "r" 인 자료와 cyl이 5인 자료 제외

```
〉 pp 〈– mpg %〉% filter(drv!="r" & cyl!=5) %〉% ggplot(mapping=aes(x=displ,y=hwy)) + geom_point()
〉 pp + facet_grid(drv~.)
〉 pp + facet_grid(~cyl)
〉 pp + facet_grid(drv~cyl)
```



연속형 변수에 의한 faceting

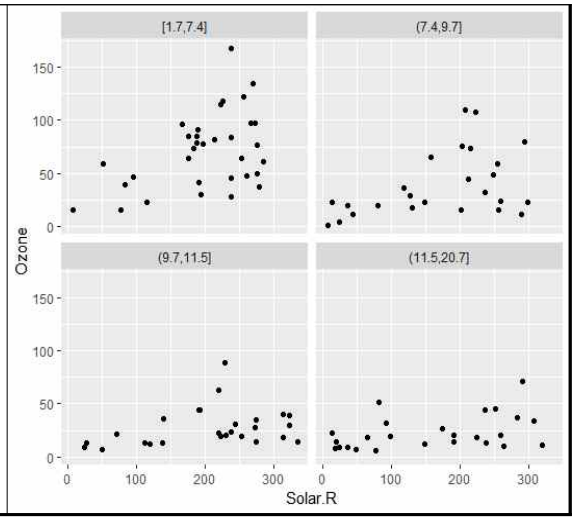
```
# 연속형 변수를 범주형 변수로 변호나 후 faceting

# 유용한 함수
 – cut_interval(x,n) : 벡터 x를 n개의 같은 길이의 구간으로 구분
 – cut_width(x,n) : 벡터 x를 길이가 witdh인 구간으로 구분
 – cut_number(x,n) : 벡터 x를 n개의 구간으로 구분하되 각 구간에 속한 데이터의 개수가 비슷하게 구분.
```

airquality에서 변수 Ozone, Solar.R, Wind의 관계탐색

(1) 변수 Wind를 4개의 구간으로 구분하되 속한 자료의 개수가 비슷하도록

(2) 4개의 구간에서 Ozone과 Solar.R의 산점도를 작성

```
〉 airquality %〉% as.tibble() %〉% mutate(Wind_R=cut_number(Wind,4)) %〉%
+    ggplot(mapping=aes(x=Solar.R,y=Ozone)) +
+    geom_point() +
+    facet_wrap(~Wind_R)

# 변수 Wind가 큰 값을 가질수록 두 변수의 관계는 점점 미약해지고 있다.
# 세 연속현 변수의 관계 탐색 방법 중 하나이다.
```

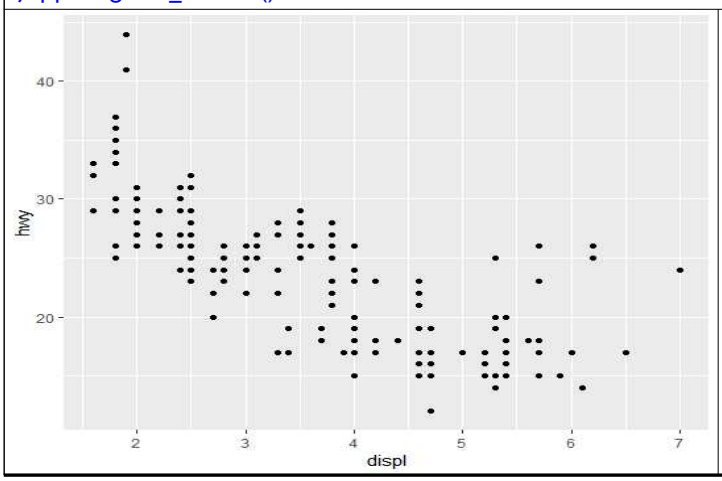


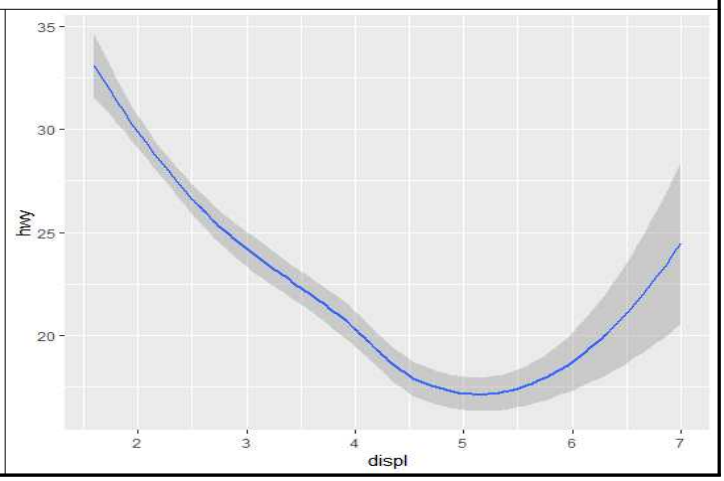
기하 객체 : Geometric object

```
# Base graphics에서 그래프 작성 방식 : pen on paper
 – 높은 수준의 그래프 함수 : 좌표축과 주요 그래프 작성
 – 낮은 수준의 그래프 함수 : 점, 선, 문자 등을 추가하여 원하는 그래프 작성

# ggplot2에서 그래프 작성 방식
 – 작성하고자 하는 그래프 : 몇몇 유형의 그래프 (점, 선 등등)
 – 몇몇 유형의 그래프를 각기 따로 작성
 – 작성된 그래프를 겹쳐 놓음으로써 원하는 그래프 작성

# ggplot2 시스템
 – 원하는 유형의 그래프 작성 : 해당되는 기하 객체(geom)를 사용하여 그래프 작성

# 기하 객체의 사용
 – 해당되는 geom 함수의 실행
 – geom 함수 실행 –〉 해당 유형의 그래프가 작성된 layer 생성
 – 여러 개의 geom 함수 실행 : 여러 layer 생성되고 이것들이 겹쳐져서 원하는 그래프 완성
```

동일 자료에 다른 geom 적용

```
# mpg의 변수 displ과 hwy를 대상으로 point geom 과 smooth geom 적용
 – point geom : 점 그래프 작성
 – smooth geom : 비모수 회귀곡선 작성

〉 pp 〈– mpg %〉% ggplot(mapping=aes(x=displ,y=hwy))
〉 pp + geom_point()
〉 pp + geom_smooth()
```





geom 함수 리스트

```
# 현재 대략 30개 이상의 geom 함수가 있다.

# 한 변수에 대한 함수 : geom_bar(), geom_histogram(), geom_density(), geom_dotplot() 등등

# 두 변수에 대한 함수 : geom_point(), geom_smooth(), geom_text(), geom_line(), geom_boxplot() 등등

# 세 변수에 대한 함수 : geom_contour(), geom_tile() 등등

# geom 함수의 리스트 : R studio의 메뉴에서 "help –〉 Cheatsheets –〉 Data Visaulization with ggplot2" 에서 확인가능하다.
```

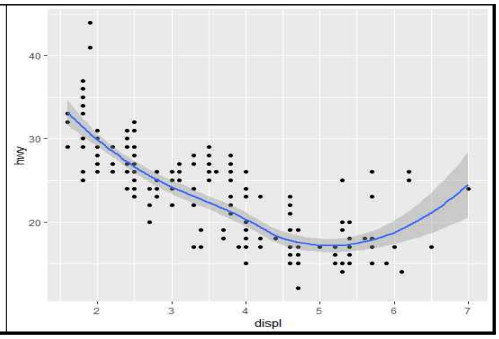
글로벌 매핑과 로컬 매핑

- 글로벌 매핑 : 함수 ggplot()에서의 매핑. 해당 그래프 작성에 필요한 모든 geom 함수에 적용
- 로컬 매핑 : geom 함수에서의 매핑. 해당 geom 함수로 작성되는 layer에만 적용. 해당 layer에서 글로벌 매핑보다 우선해서 적용
- ggplot(data, mapping=aes()) + 글로벌 매핑
  geom_*(mapping=aes()) + 로컬 매핑
  geom_*(mappinge=aes())

mpg의 변수 displ과 hwy의 산점도에 비모수 회귀곡선 추가

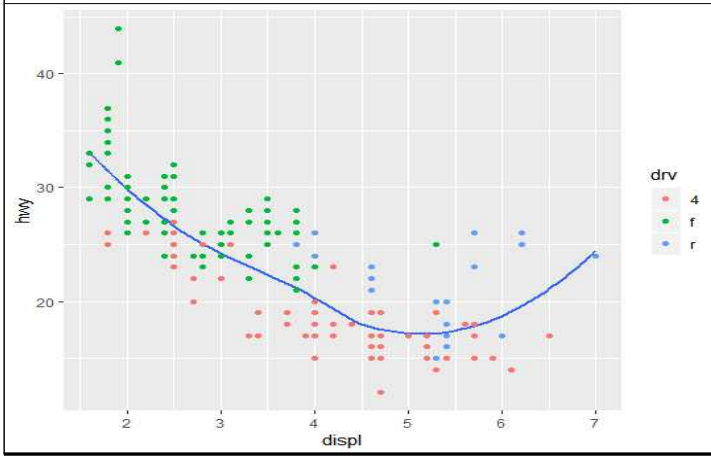
글로벌 매핑을 사용하여 중복을 피할 수 있다.

```
〉 mpg %〉% ggplot(mapping=aes(x=displ,y=hwy)) +
+    geom_point() +
+    geom_smooth()
```

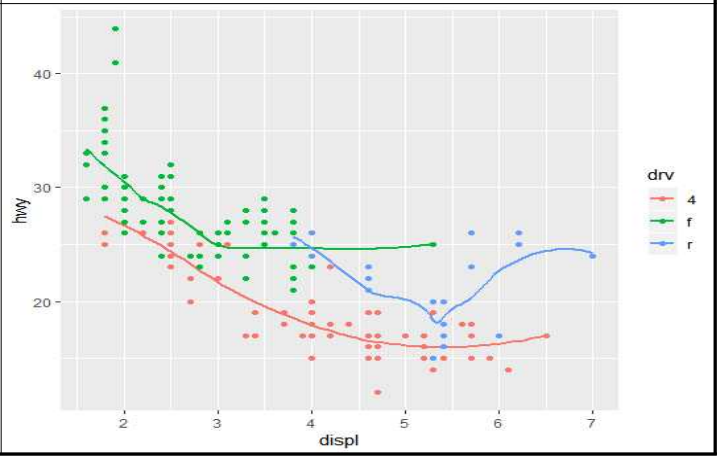


mpg의 변수 displ과 hwy의 산점도에 비모수 회귀곡선 작성, 그 위에 산점도 추가하되 drv 값에 따라 점의 색을 구분.

```
〉 # 비모수 회귀곡선은 한 개만 그림.
〉 mpg %〉% ggplot(mapping=aes(x=displ,y=hwy)) +
+    geom_smooth(se=FALSE) +
+    geom_point(mapping=aes(color=drv))
```

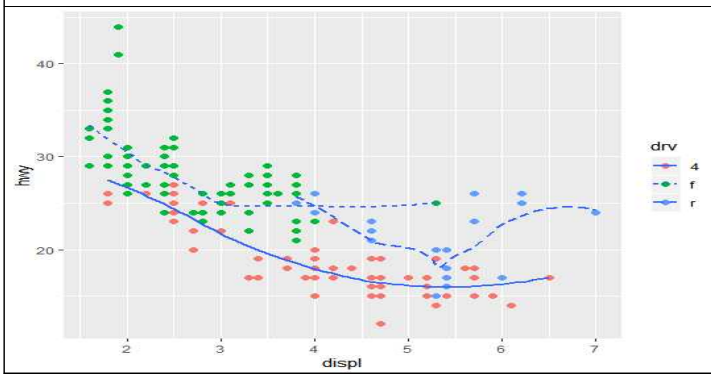



```
〉 # 비모수 회귀곡선을 drv에 따라 그림.
〉 mpg %〉% ggplot(mapping=aes(x=displ,y=hwy,color=drv)) +
+    geom_smooth(se=FALSE) +
+    geom_point()
```

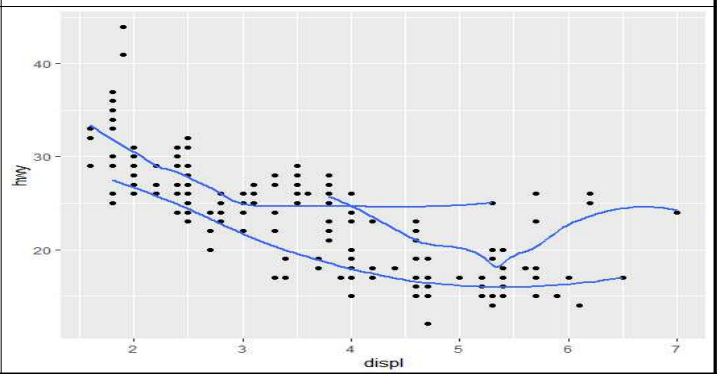


회귀곡선 작성(drv에 따라 선을 다르게 표시, 색은 같게), 산점도 작성(drv에 따라 점의 색 구분, 점 크기 확대).

```
〉 mpg %〉% ggplot(mapping=aes(x=displ,y=hwy)) +
+    geom_point(mapping=aes(color=drv),size=2) +
+    geom_smooth(mapping=aes(linetype=drv),se=FALSE)
```




```
# drv의 그룹별로 비모수 회귀곡선 작성, 선의 종류는 같은 것.
〉 mpg %〉% ggplot(mapping=aes(x=displ,y=hwy)) +
+    geom_point() +
+    geom_smooth(mapping=aes(group=drv),se=FALSE)
```

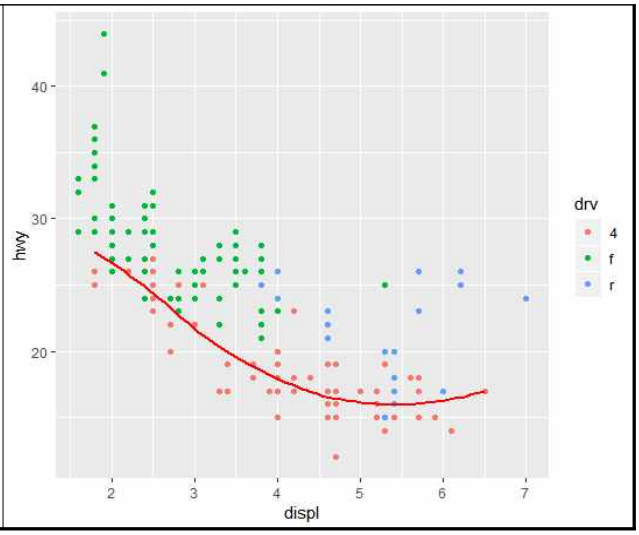


각 geom 함수에서 다른 데이터 사용

각 geom 함수로 작성되는 layer 마다 다른 데이터로 그래프 작성 가능

mpg의 변수 displ과 hwy의 산점도. drv에 따라 점의 색 구분.
비모수 회귀곡선 추가하되 drv가 4인 데이터만을 대상으로 추정

```
〉 mpg %〉% ggplot(mapping=aes(x=displ,y=hwy)) +
+    geom_point(mapping=aes(color=drv)) +
+    geom_smooth(data=filter(mpg,drv=="4"),se=FALSE, color="red")
```



통계적 변환 : Statistical transformation

그래프 작성에 사용되는 자료
(1) 입력된 자료 : 산점도
(2) 입력된 자료를 대상으로 통계적 변환 과정을 거쳐 생성된 자료 : 비모수 회귀곡선 그래프

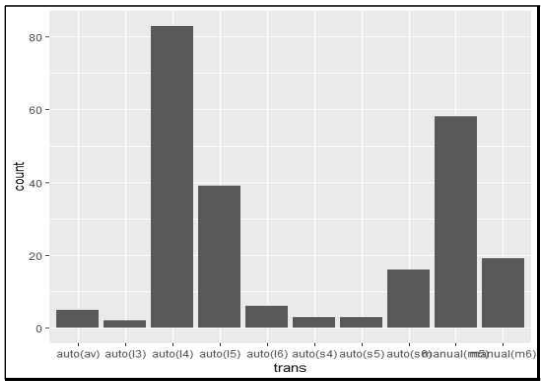
통계적 변환(stat)
- 입력된 데이터 프레임 자료의 변환을 의미
- 각 그래프 유형별 대응되는 stat 존재
  산점도 : stat = "identity"
  비모수 회귀곡선 : stat = "smooth"
  막대 그래프 : stat = "count"
- 각 geom 함수마다 대응되는 디폴트 stat 존재
  geom_point() : geom_point(stat= "identity")
  geom_smooth() : geom_smooth(stat= "smooth")
  geom_bar() : geom_bar(stat= "count")

stat 함수

geom 함수 대신 stat 함수로 그래프 작성 가능

각 stat 함수마다 디폴트 geom 존재
- stat_identity() : stat_identity(geom= "point")
- stat_smooth() : stat_smooth(geom= "smooth")
- stat_count() : stat_count(geom= "bar")

예 : mpg의 변수 trans의 막대 그래프 작성



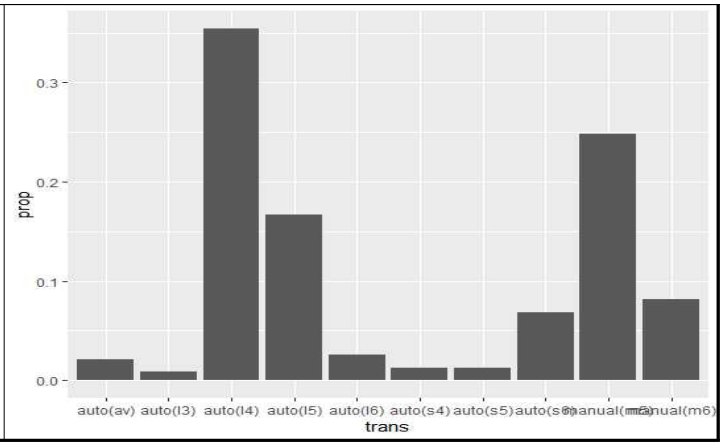

```
# geom 함수
〉 mpg %〉% ggplot(mapping=aes(x=trans)) +
+    geom_bar()
# stat 함수
〉 mpg %〉% ggplot(mapping=aes(x=trans)) +
+    stat_count()
```

stat 함수로 계산된 변수의 이용

```
# stat 함수 : 입력된 데이터 프레임을 대상으로 변환을 실시하여 그래프 작성에 필요한 변수로 이루어진 데이터 프레임을 내부적으로 생성, 생성된 변수를 사용자가 직접 지정해서 사용 가능

# 예 : geom_bar() 혹은 stat_count()에서 계산된 변수
        – 변수 count : 각 범주의 빈도
        – 변수 prop : 그룹별 비율

# 계산된 변수를 사용자가 지정할 때에는 변수를 ".." 기호를 감싸야 한다. 기존의 변수와 혼동을 방지.
```

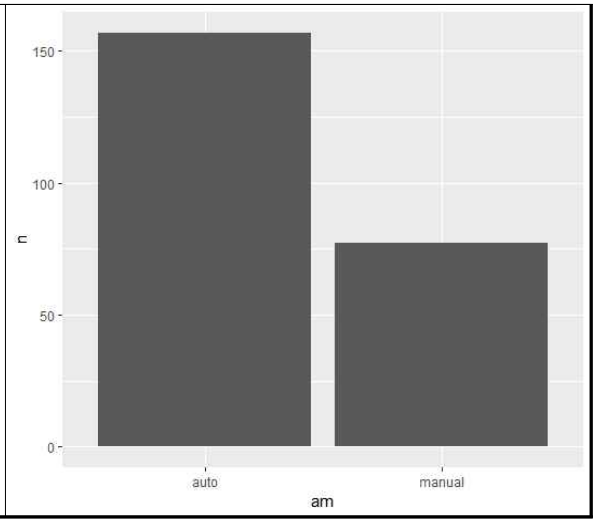
mpg의 변수 trans의 막대 그래프를 상대 도수로 작성

```
# 모든 범주가 각각의 그룹으로 구성되어 있기 때문에
–〉 group= "아무거나" 로 하나의 그룹으로 묶어주어야 한다.

〉 mpg %〉% ggplot() +
+    geom_bar(mapping=aes(x=trans,y=..prop..,group=1))
```



geom 함수에서 stat을 따로 지정해야 하는 경우

```
# geom 함수의 디폴트 stat이 아닌 다른 stat을 사용해야 하는 경우
# 예) 도수분포표로 막대그래프를 작성하는 경우

# mpg의 trans를 auto와 manual로 통합. 도수분포표 작성. 막대그래프 작성.

〉 mpg %〉% mutate(am=substr(trans,1,nchar(trans)–4)) %〉%
+    group_by(am) %〉%
+    summarize(n=n()) %〉%
+    ggplot() +
+    geom_bar(mapping=aes(x=am,y=n),stat="identity")
```



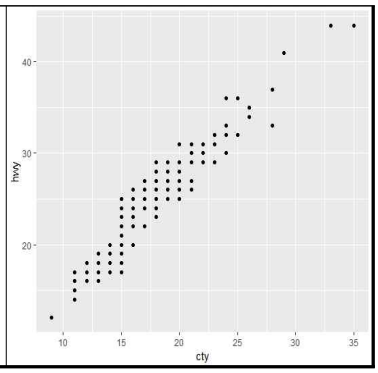
위치 조정 : Position adjustments

```
# 그래프 요소들의 위치 조정
 – 연속형 자료 : 산점도의 점이 겹쳐지는 경우
 – 범주형 자료 : 이변량 막대 그래프 작성

# 산점도의 점이 겹치는 문제
 – 산점도 작성의 가장 큰 문제
 – 해결방안 : 반올림 처리 등으로 같은 값이 많은 자료의 경우 : 자료에 약간의난수를 더해 점의 위치 조정 (jittering)
              대규모의 자료가 좁은 구역에 몰려서 한 무리를 형성하는 경우 : 추후에 다룰 예정
# 이변량 막대 그래프
 – 쌓아올린 막대 그래프 / 옆으로 붙여 놓은 막대 그래프 / mosaic plot
```

산점도에서 점이 겹치는 문제 해결

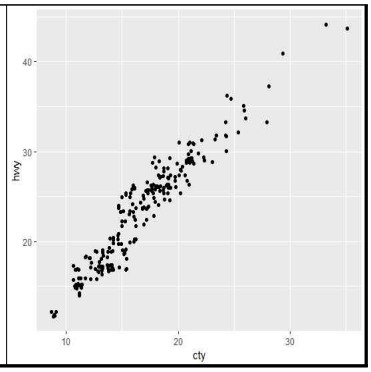
```
# mpg에서 변수 cty와 hwy의 산점도
〉 ggplot(data=mpg) +
+    geom_point(mapping=aes(x=cty,y=hwy))

# 산점도에 나타난 점의 개수가 전체 데이터 개수인 234개에
훨씬 못 미쳐 보인다.
# 두 변수의 값이 반올림 처리되어 같은 값이 많아짐.
# jittering : 자료에 약간의 난수 추가
```

$$(x,y) \rightarrow (x+\varepsilon, y+\varepsilon) \qquad \varepsilon \sim Unif(-\alpha, \alpha)$$



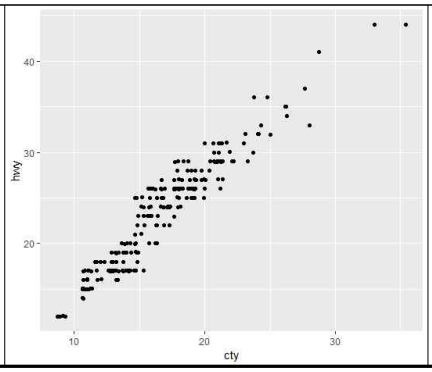
jittering 실시

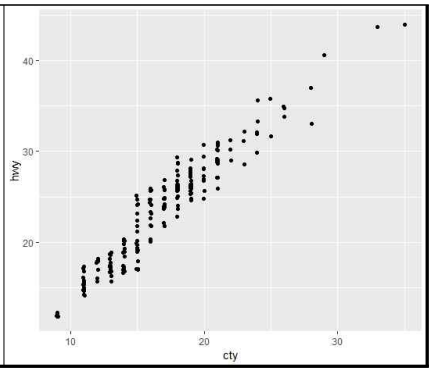
```
〉 ggplot(data=mpg) +
+    geom_point(mapping=aes(x=cty,y=hwy),position="jitter")

# 작성되는 그래프마다 미세한 차이가 발생
# 추가되는 난수의 크기를 조절하고자 하는 경우에는 함수 geom_jitter() 사용
```



함수 geom_jitter() 사용

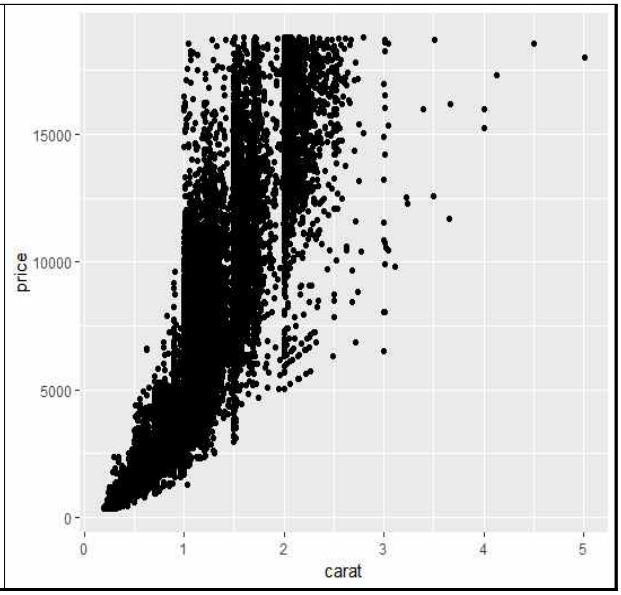
```
〉 ggplot(data=mpg,mapping=aes(x=cty,y=hwy)) +
+    geom_jitter(width=0.4,height=0.05)

〉 ggplot(data=mpg,mapping=aes(x=cty,y=hwy)) +
+    geom_jitter(width=0.05,height=0.4)
```





산점도로 해결이 불가능한 경우

```
# diamonds에서 변수 carat과 price의 산점도
〉 ggplot(data=diamonds) +
+    geom_point(mapping=aes(x=carat,y=price))

# 너무 많은 점들이 밀집되어 있는 상황

# 두 변수의 정확한 관계 파악이 어려운 그래프

# jittering으로는 해결이 불가능하다.
```

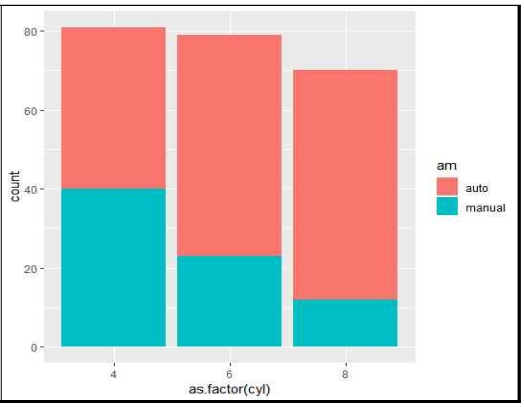


이변량 막대 그래프

```
# 막대 그래프 작성 : geom-bar()
# 이변량 막대 그래프 : 함수 geom_bar()에 시각적 요소 x와 fill, position 사용
```

mpg에서 trans의 범주를 auto와 manual로 통합한 변수 am 생성. 변수 cyl이 5인 케이스 제거 후 am과 cyl의 이변량 막대그래프

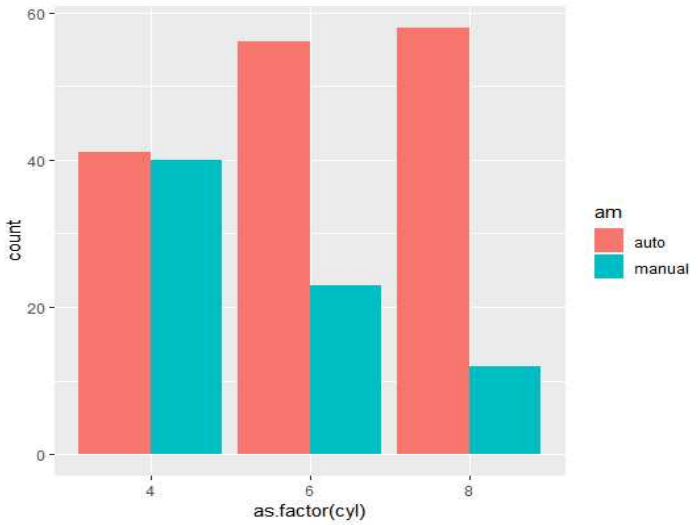
```
〉 mpg %〉% filter(cyl!="5") %〉% mutate(am=substr(trans,1,nchar(trans)-4)) %〉%
+    ggplot() +
+    geom_bar(mapping=aes(x=as.factor(cyl),fill=am))
```



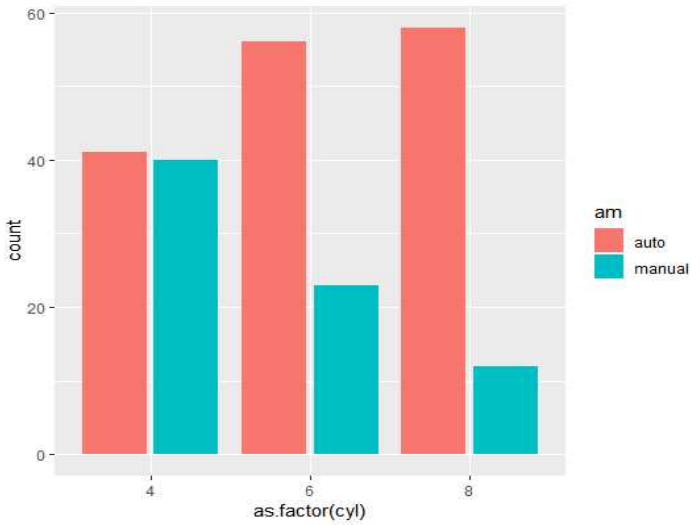
ggplot() + geom_bar(mapping=aes(x=cyl,fill=am))
-〉 x축이 3,4,5,6,7,8 로 생성된다.

이변량 막대 그래프

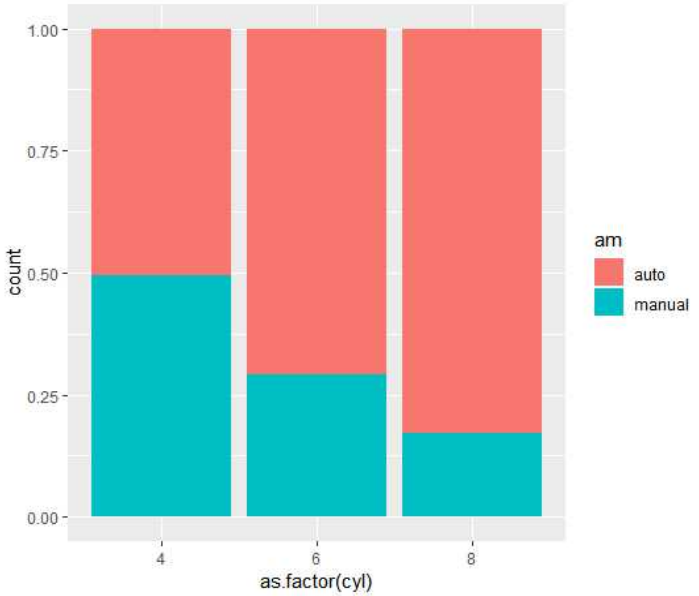
```
mpg %〉% filter(cyl!="5") %〉% mutate(am=substr(trans,1,nchar(trans)-4)) %〉%
  ggplot() +
  geom_bar(mapping=aes(x=as.factor(cyl),fill=am),position="dodge")
```




```
mpg %〉% filter(cyl!="5") %〉% mutate(am=substr(trans,1,nchar(trans)-4)) %〉%
  ggplot() +
  geom_bar(mapping=aes(x=as.factor(cyl),fill=am),position="dodge2")
```




```
mpg %〉% filter(cyl!="5") %〉% mutate(am=substr(trans,1,nchar(trans)-4)) %〉%
  ggplot() +
  geom_bar(mapping=aes(x=as.factor(cyl),fill=am),position="fill")
```



position = "stack" (디폴트값)
- 〉 위로 쌓아 올린 막대 그래프

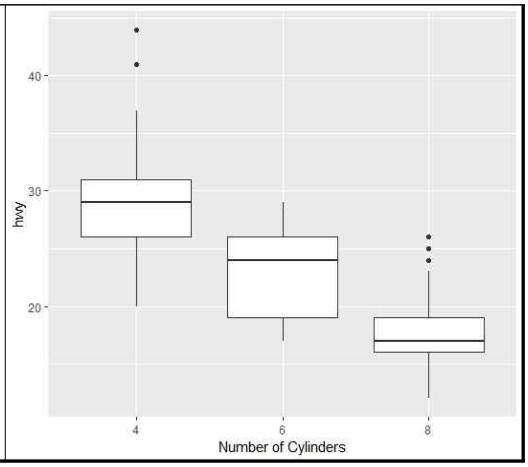
position = "dodge"
- 〉 옆으로 쌓아 올린 막대 그래프

position = "dodge2"
- 〉 옆으로 살짝 떨어진 옆으로 쌓아 올린 막대 그래프

position = "fill"
- 〉 조건부확률로 위로 쌓아 올린 막대 그래프

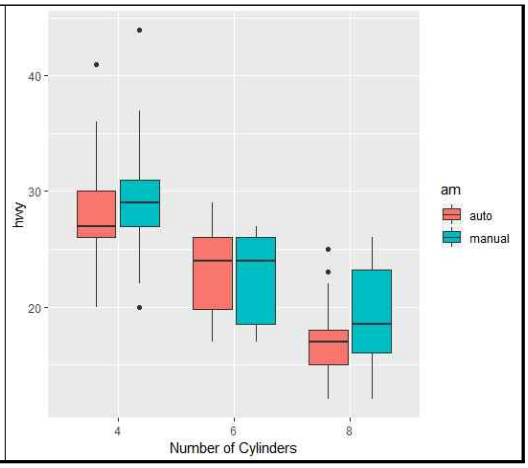
나란히 서있는 상자그림

```
# geom_boxplot()
# 필요한 시각적 요소 : x = 그룹을 구성하는 변수(요인)
                       y = 연속형 변수

# mpg에서 cyl=5 케이스를 제거하고 auto와 manual를 통합한 뒤,
# cyl 그룹에 따른 hwy의 상자그림
> mpg %>% filter(cyl!="5") %>% mutate(am=substr(trans,1,nchar(trans)-4)) %>%
+    ggplot(mapping=aes(x=as.factor(cyl),y=hwy)) +
+    geom_boxplot() +
+       xlab("Number of Cylinders")
```



그룹을 구성하는 변수가 2 개인 경우의 상자그림

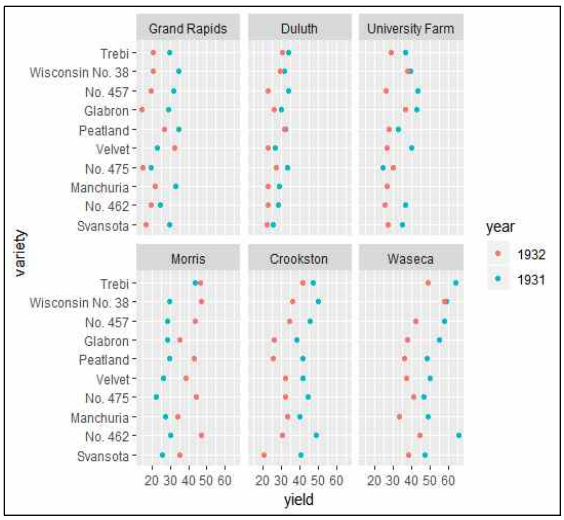
```
# 변수 am에 따라 다른 색이 채워져 있고, 두 상자그림이 옆에 붙어 있다.
# 필요한 시각적 요소 : x, y, fill, position
# postion= "dodge" 가 디폴트로 적용됨.

> mpg %>% filter(cyl!="5") %>% mutate(am=substr(trans,1,nchar(trans)-4)) %>%
+    ggplot(mapping=aes(x=as.factor(cyl),y=hwy)) +
+    geom_boxplot(mapping=aes(fill=am)) +
+       xlab("Number of Cylinders")
```



연습문제 1

패키지 lattice에 있는 데이터 프레임 barley는 미네소타 주 농경학자들이 보리 종류에 따른 수확량의 차이를 비교하기 위해 2년간 경작하여 얻은 자료이다. 설명변수로는 6군데 경작지(site), 10종류의 보리(variety), 경작 년도(year)이고 반응변수는 수확량(yield)이다. 세 설명변수의 조합에 따른 수확량의 분포를 알아보는 그래프를 작성해 보자.

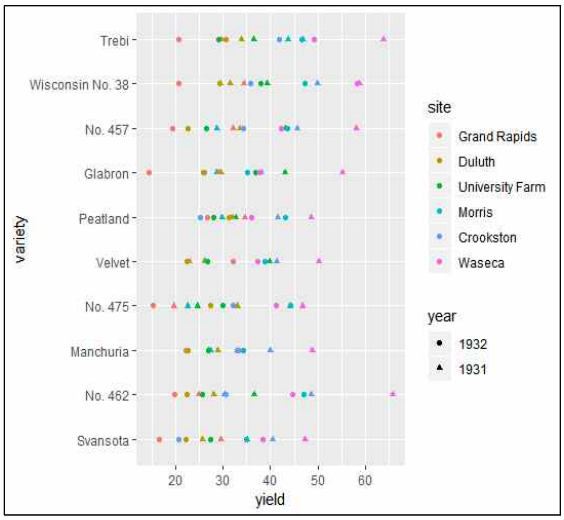
```
lattice::barley %>% as.tibble() %>%
  ggplot(mapping=aes(x=yield,y=variety,color=year)) +
  geom_point() +
  facet_wrap(~site)
```



연습문제 2

보리 종류(variety)에 따른 수확량(yield)의 비교분석에서 경작지(site)와 년도(year)의 효과를 단순 반복으로 처리한 다음의 그래프를 작성해보자.

```
lattice::barley %>% as.tibble() %>%
  ggplot(mapping=aes(x=yield,y=variety,color=site)) +
  geom_point(mapping=aes(shape=year))
```



연습문제 3

보리 종류(variety)의 평균 수확량을 계산하여 다음과 같이 크기 순으로 나타내자.

```
lattice::barley %>% as.tibble() %>%
  group_by(variety) %>%
  summarise(mean_yield=mean(yield)) %>%
  arrange(desc(mean_yield))
```

```
# A tibble: 10 x 2
   variety          mean_yield
   <fct>                 <dbl>
 1 Trebi                  39.4
 2 Wisconsin No. 38       39.4
 3 No. 457                35.8
 4 No. 462                35.4
 5 Peatland               34.2
 6 Glabron                33.3
 7 Velvet                 33.1
 8 No. 475                31.8
 9 Manchuria              31.5
10 Svansota               30.4
```

연습문제 4

mpg의 변수 hwy는 자동차의 고속도로 연비를 나타낸다. 범주형 변수인 fl(연료 종류), trans(변속기 종류)에 따른 hwy의 분포를 알아보자. 변수 fl의 종류별 빈도를 구하라.

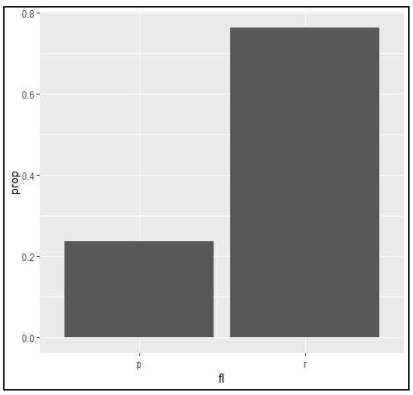
```
mpg %>% group_by(fl) %>%
  summarise(n=n())
```

```
# A tibble: 5 x 2
  fl        n
  <chr> <int>
1 c         1
2 d         5
3 e         8
4 p        52
5 r       168
```

연습문제 5

변수 fl에서 c,d,e,는 제외하고 p와 r만을 대상으로 상대도수에 의한 막대 그래프를 작성해 보자.

```
mpg %>% mutate(am=substr(trans,1,nchar(trans)-4)) %>%
  filter(fl=="r"|fl=="p") %>%
  ggplot() +
  geom_bar(mapping=aes(x=fl,y=..prop..,group=1))
```



연습문제 6

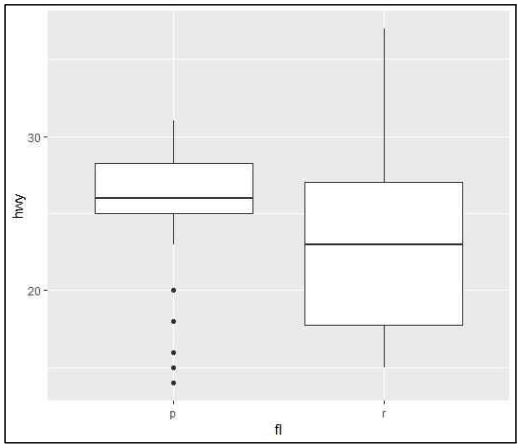
변수 trans의 종류별 빈도를 다음과 같이 구하라.

```
mpg %>% group_by(trans) %>%
  summarise(n=n())
```

```
# A tibble: 10 x 2
   trans        n
   <chr>    <int>
 1 auto(av)     5
 2 auto(l3)     2
 3 auto(l4)    83
 4 auto(l5)    39
 5 auto(l6)     6
 6 auto(s4)     3
 7 auto(s5)     3
 8 auto(s6)    16
 9 manual(m5)  58
10 manual(m6)  19
```

연습문제 7

```
# 변수 fl에 따른 hwy의 분포를 상자그림으로 나타내 보자.

mpg %>% filter(fl %in% c("p","r")) %>%
  ggplot() +
  geom_boxplot(mapping=aes(x=fl,y=hwy))
```



연습문제 8

```
# 변수 am과 fl에 따른 hwy의 분포 비교를 위해 상자그림을 작성해 보자.

mpg %>% filter(fl %in% c("p","r")) %>%
  mutate(am=substr(trans,1,nchar(trans)-4)) %>%
  ggplot(mapping=aes(x=fl,y=hwy)) +
  geom_boxplot() +
  facet_wrap(~am,ncol=1)
```